Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт электронных и информационных систем

Кафедра информационных технологий и систем

**ОТЧЕТ**

по лабораторной работе № 6

“Обработка списков”

Разработал:

Студент группы 0092

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Потапенко Е.А

«\_\_\_\_»\_\_\_\_\_\_\_\_2023 г.

Проверил преподаватель:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Михайлов Д.В.

«\_\_\_\_»\_\_\_\_\_\_\_2023 г.

Великий Новгород

2023

1. Цели и задачи проводимого исследования  
     
   Целью работы является изучение приемов работы со списками в Прологе, а также более детальное изучение рекурсивного программирования применительно к обработке списков.
2. Задание 1  
     
   Написать программу сортировки списка методом Шелла на Прологе. Вычисление последовательности шагов сортировки производится в соответствии с методом Дональда Кнута  
     
   Решение:  
     
   Код, реализующий задачу, получился слишком длинным для этого документа, его можно найти в приложенном файле **code.txt**  
     
   Объяснение основных функций:  
     
   **Генерация последовательности Шелла**Изображение выглядит как текст, Шрифт, снимок экрана, белый

   Автоматически созданное описание **shell\_sequence** создает последовательность шагов для сортировки Шелла, основываясь на размере списка **Len**.

**initial\_h** вычисляет начальное значение шага **H** согласно методу Шелла.

**generate\_sequence** строит фактическую последовательность шагов.  
  
  
**Сортировка Шелла  
Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание  
shell\_sort** вызывает **shell\_sequence** для создания последовательности шагов и затем передает список и эту последовательность предикату **shell\_pass**. **Подготовка списка для сортировки  
Изображение выглядит как текст, Шрифт, белый, линия

Автоматически созданное описание  
  
shell\_pass** проходит по всей последовательности шагов и сортирует списки с использованием предиката **shell\_sublist\_pass.  
  
Сортировка подсписков  
Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание  
  
shell\_sublist\_pass** разделяет список на подсписки с шагом H с помощью **partition** и затем объединяет их с помощью **merge\_sublists.  
  
Объединение подсписков  
Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание  
merge\_sublists** используется для объединения отсортированных подсписков в один отсортированный список.  
  
**Другие вспомогательные предикаты:  
  
initial\_h**, **generate\_sequence** и другие вспомогательные предикаты используются для вычисления шагов Шелла и разделения списка на подсписки для сортировки.

Каждый предикат выполняет свою часть работы по вычислению шагов Шелла, разделению и слиянию списков для достижения сортировки Шелла.  
  
**Результат работы:  
Изображение выглядит как текст, Шрифт, снимок экрана, линия

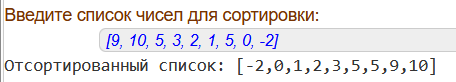
Автоматически созданное описание**

1. Задание 2  
   Написать сортировку методом пузырька  
     
   Решение:  
     
   Изображение выглядит как текст, снимок экрана, Шрифт, число

   Автоматически созданное описание  
     
   **bubble\_sort** - основной предикат сортировки пузырьком.

**swap** - предикат, который меняет местами элементы в списке, если они находятся в неправильном порядке.

**main** - предикат, который запрашивает у пользователя ввод списка чисел и выводит отсортированный список с использованием предиката **bubble\_sort**.

Этот код принимает список чисел от пользователя, сортирует его методом пузырька и выводит отсортированный результат.  
  
**Результат работы:**

1. Задание 3  
   Написать программу решения головоломки.  
   «Числа по периметру».

* По периметру пятиконечной звезды необходимо проставить числа от 1 до 10 так, чтобы суммы чисел в концах любого отрезка не делились ни на 3, ни на 5, ни на 7.  
    
  Решение:  
  Изображение выглядит как текст, снимок экрана, Шрифт, число

  Автоматически созданное описание  
    
  Используется библиотека **clpfd** для работы с ограничениями о доменах переменных.  
    
  **Nums** - список переменных, представляющих числа на вершинах пятиконечной звезды.

**Nums** **ins 1..10** - определение домена переменных от 1 до 10.

**all\_distinct(Nums)** - все числа в списке должны быть различными.

**valid\_sums([A, B, C, D, E])** - вызов предиката для проверки сумм на рёбрах.

**labeling([], Nums)** - поиск решения.

**Sum1..Sum5** - переменные для хранения сумм на рёбрах звезды.

**all\_valid([Sum1, Sum2, Sum3, Sum4, Sum5])** - вызов предиката для проверки, удовлетворяют ли все суммы условиям задачи.

**all\_valid** проверяет, удовлетворяют ли все суммы условиям **not\_divisible\_by\_357.  
  
not\_divisible\_by\_357 -** Проверяет, что сумма не делится на 3, 5 или 7.  
  
**Результат работы:  
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**Код создаёт переменные для чисел на вершинах звезды, определяет суммы на рёбрах и проверяет, удовлетворяют ли эти суммы условиям задачи. Если все условия выполнены, программа находит комбинацию чисел, удовлетворяющую заданным условиям.

1. Задание 4  
     
   Написать программу вставки подсписка в список, начиная с i-го элемента  
     
   Решение:  
   Изображение выглядит как текст, снимок экрана, Шрифт, число

   Автоматически созданное описание  
     
   Результат работы:  
   Изображение выглядит как текст, снимок экрана, Шрифт, линия

   Автоматически созданное описание  
     
   Изображение выглядит как текст, снимок экрана, Шрифт, линия

   Автоматически созданное описание
2. Вывод  
   Лабораторная работа выполнена в полном объеме. В ее ходе я изучил приемы работы со списками в Прологе, а также более детальное изучение рекурсивного программирования применительно к обработке списков.